CSS GRID

Mise en place d'une grille

Une grille nous permet d'organiser des éléments sur une page, en fonction des régions créées par les guides. (lignes horizontales et verticales). On définit ainsi des colonnes et des rangées entre ces lignes et des goutières (**gap**) entre elles.

Une aire (**grid area**) est définie par plusieurs cellules adjacentes. Pour créer notre première grille ci-dessus, nous avons besoin d'un élément conteneur, ici de classe `grid-1` et de neuf enfants.

```
<section class="grid-1">
  <div class="item-1">1</div>
  <div class="item-2">2</div>
  <div class="item-3">3</div>
  <div class="item-4">4</div>
  <div class="item-5">5</div>
  <div class="item-6">6</div>
  <div class="item-7">7</div>
  <div class="item-8">8</div>
  <div class="item-9">9</div>
</section>
```

Un peu de style pour différentier le tout :

```
grid-1 div {
  color: white;
  font-size: 20px;
  padding: 20px;
  }

/* specific item styles */

.item-1 {
  background: #b03532;
}
.item-2 {
  background: #33a8a5;
}
.item-3 {
  background: #30997a;
}
.item-4 {
  background: #6a478f;
}
.item-5 {
  background: #da6f2b;
}
```

```
.item-6 {
  background: #3d8bb1;
}
.item-7 {
  background: #e03f3f;
}
.item-8 {
  background: #59a627;
}
.item-9 {
  background: #4464a1;
}
```

Règles de la grille

Tout d'abord, nous devons déclarer que notre élément conteneur est une grille utilisant une nouvelle valeur pour la propriété `display` :

```
.grid-1 {
  display: grid;
}
```

Ensuite, nous devons définir notre grille, en indiquant le nombre de pistes qu'elle aura, horizontalement et verticalement. Nous faisons cela avec les propriétés `grid-template-columns` et `grid-template-rows`:

```
.grid-1 {
  display: grid;
  grid-template-columns: 150px 150px 150px;
  grid-template-rows: auto auto auto;
}
```

Les hauteurs de lignes sont auto par défaut, prenant la hauteur du contenu.

Ajoutez des goutières

Nous pouvons utiliser `grid-column-gap` et `grid-row-gap`, ou `grid-gap` propriété abrégée .

Ajoutons une gouttière fixe de 20px à notre .grid-1 élément.

```
.grid-1 {
  display: grid;
  grid-template-columns: 150px 150px 150px;
  grid-template-rows: auto auto auto;
  grid-gap: 20px;
}
```

Unités flexibles

L'objectif de Grid est de nous permettre de contrôler correctement la mise en page sur le Web. Nous allons donc fluidifier notre grille statique avant d'aller plus loin.

Il est déjà tout à fait possible d'utiliser d'autres unités ici, comme `em` ou `rem` par exemple. Ou encore des unités comme `vh` et `vmin` . Dans ce cas, nous allons changer nos unités de pixels pour les pourcentages.

```
.grid-1 {
  display: grid;
  grid-template-columns: 33% 33% 33%;
  grid-template-rows: auto auto auto;
  grid-gap: 20px;
}
```

Répéter

Écrivons ceci est une manière plus ordonnée, en utilisant la fonction `repeat()` :

```
grid-template-columns : repeat(3, 33.33%);
```

L'unité fr

Une dernière amélioration peut être apportée à notre grille simple et elle résoudra le problème de largeur des gap; nous allons introduire l' unité `fr` ou fraction . Une seule unité `fr` décrit «un élément parmi tous les éléments dans lesquels nous le divisons» . Par exemple, nous pourrions déclarer nos colonnes en utilisant:

```
grid-template-columns: 1fr 1fr 1fr;
```

Ici, il y a un total de trois unités fr, donc chaque colonne aura un tiers de large. Voici un autre exemple:

```
grid-template-columns: 2fr 1fr 1fr
```

Il y a maintenant quatre unités `fr`, la première colonne occuperait la moitié de la largeur disponible, les deux autres colonnes représentant chacune un quart.

Ces unités sont vraiment puissantes, surtout en combinaison avec d'autres unités de mesure:

```
grid-template-columns: 300px 1fr 3fr 20%;
```

Ici, nous avons déclaré quatre colonnes:

- la première est fixé à 300px de large
- la dernière est un élément flexible de 20% de la largeur du conteneur
- les unités fr sont calculées en tenant également compte des gouttières, en laissant la deuxième colonne avec une partie de l'espace restant et le troisième avec trois pièces.

Définir des zones de grille

Testons :

```
.item-1 {
  background: #b03532;
  grid-column: 1 / 3;
  grid-row: 1;
}
```

Dans cette déclaration abrégée `grid-column` , nous utilisons implicitement `grid-column-start` et `grid-column-end` : **l'élément va commencer à la ligne 1 de la grille et finir à la ligne 3**.

On remarque le placement automatique pour les autres éléments de Grid.

La même chose peut être faite avec des lignes, ce qui nous donnerait une zone beaucoup plus grande en haut à gauche de notre grille.

```
.item-1 {
  background: #b03532;
  grid-column: 1 / 3;
  grid-row: 1 / 3;
}
```

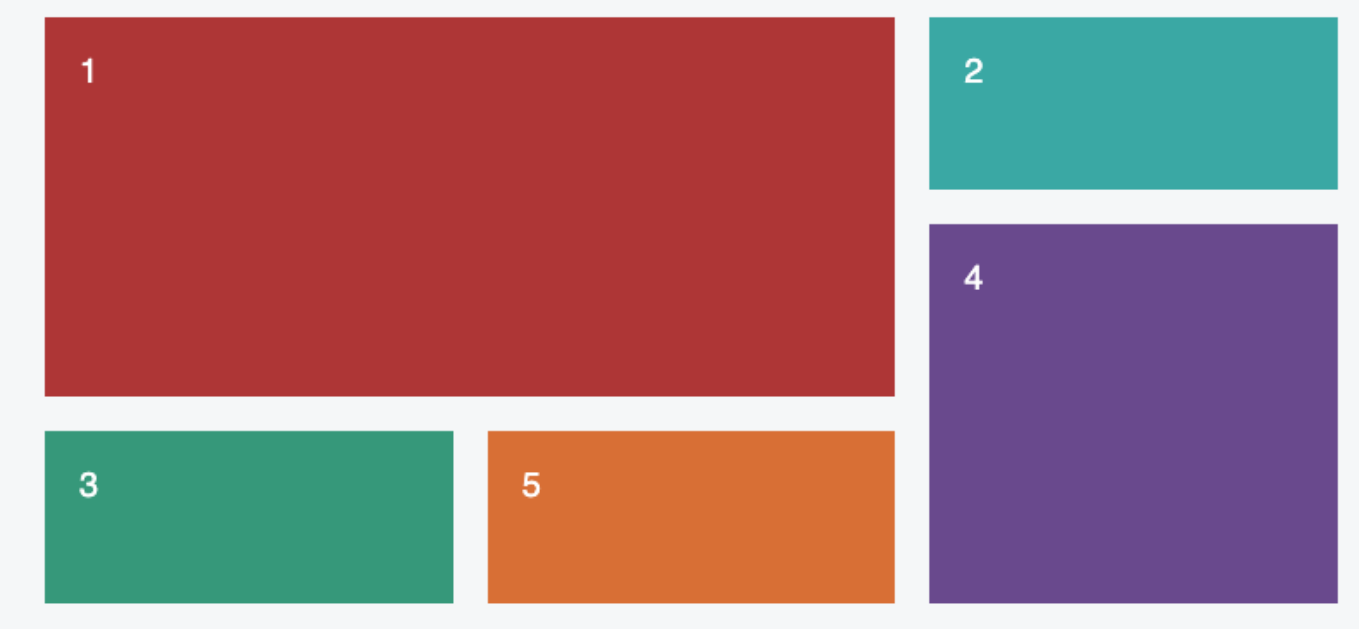
Spanning Cells

En utilisant une syntaxe peut-être plus simple, nous pouvons utiliser le mot clé `span`. Avec `span`, nous, nous ne sommes pas liés à spécifier où la zone se termine, mais à définir le nombre de pistes sur lesquelles l'élément doit être réparti:

```
.item-1 {
  background: #b03532;
  grid-column: 1 / span 2;
  grid-row: 1 / span 2;
}
```

Cela nous donne le même résultat final, mais si nous changions l'emplacement où nous voulons que l'item commence, nous ne serions plus obligés de changer la fin.

A vous de jouer :



Zones nommées

L'utilisation des méthodes de numérotation que nous avons décrites jusqu'ici fonctionne parfaitement, mais les zones de modèle de grille peuvent rendre la définition des présentations encore plus intuitive.

Plus précisément, ils nous permettent de nommer des zones sur la grille. Avec ces zones nommées, nous pouvons les référencer pour positionner nos éléments.

- en tête
- main content
- barre latérale
- bas de page

Nous définissons ces zones sur notre conteneur de grille :

```
.grid-1 {
    /* ..existing styles */

    grid-template-areas:    "header header header"
                            "main main sidebar"
                            "footer footer footer";
}
```

Positionner les objets

On abandonne `grid-column` et `grid-row` en faveur de `grid-area`:

```
.item-1 {
  background: #b03532;
  grid-area: header;
}
.item-2 {
```

```
  background: #33a8a5;
  grid-area: main;
}
.item-3 {
  background: #30997a;
  grid-area: sidebar;
}
.item-4 {
  background: #6a478f;
  grid-area: footer;
}
```

- Le premier élément est inséré dans l'en-tête et s'étend sur les trois colonnes d'en-tête.
- Le deuxième élément se voit attribuer la zone de contenu principal,
- le troisième devient notre barre latérale et le quatrième notre pied de page.

Zones de grille imbriquée

Une page Web donnée contiendra toutes sortes de composants imbriqués, voyons comment cela fonctionne avec Grid.

Lorsque nous déclarons un conteneur de grille, `display: grid;` **seuls ses descendants directs deviennent des éléments de grille** . Le contenu que nous ajoutons à ces éléments enfants ne sera absolument pas affecté par Grid, sauf indication contraire expresse.

Dans notre exemple, nous allons ajouter .item-5 , .item-6 et de .item-7 en les imbriquant dans .item-2.

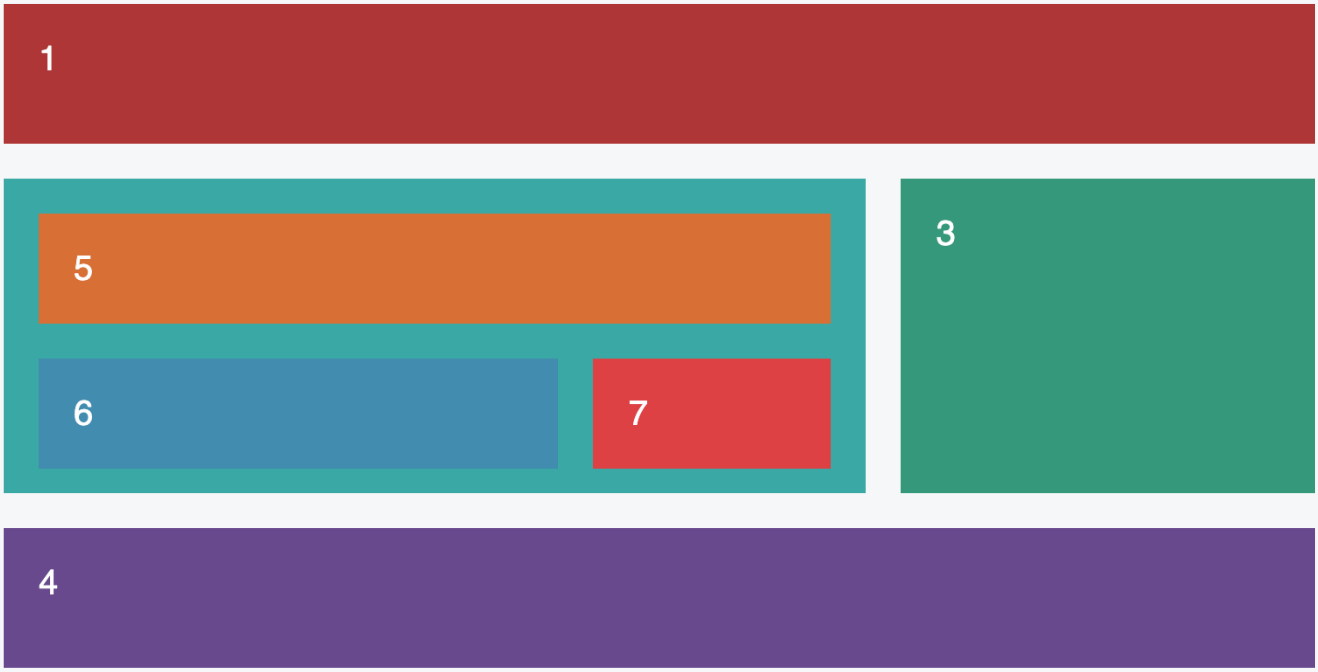
```
<section class="grid-1">
  <div class="item-1">1</div>

  <div class="item-2">
    <div class="item-5">5</div>
    <div class="item-6">6</div>
    <div class="item-7">7</div>
  </div>

  <div class="item-3">3</div>
  <div class="item-4">4</div>
</section>
```

Alors maintenant, nous devons déclarer notre .item-2 comme un conteneur de grille, en configurant sa grille avec deux colonnes et deux lignes.

```
display: grid;
grid-template-columns: 1fr 30%;
grid-template-rows: auto auto;
grid-gap: 20px;
```

```
  grid-template-areas: "header header"
                       "article sidebar";
```

Nous pouvons utiliser les noms «header» , «article» et «sidebar» à nouveau ici; il n'y a pas de confusion, car tout reste dans le contexte. Ces zones de la grille s'appliquent uniquement à la grille à l'intérieur .item-2.



Vous pouvez tester sur http://cssgridgarden.com/#fr

Récapitulatif rapide

```
    //met en place l'affichage en grille indispensable !
    display: grid;

    //détermine le nombres et les dimensions des colonnes
    grid-template-columns: 40px auto 40px;

    //détermine le nombres et les dimensions des lignes
    grid-template-rows: 40px auto 40px;

    //défini la hauteur des lignes sans définir le nombres de lignes
    grid-auto-rows: 40px;

    //espace entre les cellules
    grid-gap: 3px;

    //synthétise grid-template-rows et grid-template-columns
    //dans cet ordre.
    grid-template: repeat(2, 50px) / repeat(3, 1fr);

    //défini la tranche initiale dans la grille
    grid-column-start: 1;
```

```
    //tranche de fin. Attention ! deux cellules possédent 3 tranches !
    grid-column-end: 3;

    //version simplifier des deux propriété précédentes
    grid-column: 1 / -1;

    //Permet de créer une matrice pour la disposition des éléments
    grid-template-areas:
         "h h h h h h h h h h h h"
         "m c c c c c c c c c c c"
         "f f f f f f f f f f f f";
    //permet d'attribuer un identifiant à rappeler dans la matrice.
    grid-area: h;

    //défini une largeur minimal et rempli autant que possible l'espace.
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
```

Media-queries

```
<div class="wrapper">
  <header>L'en-tête</header>
  <nav>
    <ul>
      <li><a href="">Nav 1</a></li>
      <li><a href="">Nav 2</a></li>
      <li><a href="">Nav 3</a></li>
    </ul>
  </nav>
  <article>
    <h1>L'article principal</h1>
    <p>
       Dans cette disposition, on affiche les zones dans le même ordre que
dans le document pour les écrans dont la largeurest inférieure à 500
pixels. On passe à une disposition sur deux colonnes ou trois colonnes en
redéfinissant la grilleet le placement des objets sur la grille.
    </p>
  </article>
  <aside>Barre latérale</aside>
  <div id="ad">Publicité</div>
  <footer>Le pied de page</footer>
</div>
```

et le css : On utilise `grid-template-areas` afin de créer la disposition. On nomme les zones en dehors des différentes media queries

Les propriétés sont nommées grâce à la propriété `grid-area`.

```
header{
  grid-area: header;
}
article {
  grid-area: content;
}
nav {
  grid-area: nav;
}
aside {
  grid-area: sidebar;
}
#ad {
  grid-area: ad;
}
footer {
  grid-area: footer;
}
```

On désire avoir une seule colonne pour certaines tailles d'affichage ajouter une barre latérale lorsqu'on a plus d'espace et enfin(500px) avoir trois colonnes pour les écrans les plus larges(700px).

A vous de jouer !